10 Software Quality Factors –
That Should Always Be Remembered

Flexibility and Extensibility

Flexibility is the possibility to modify/remove functionality to software without damaging the current system. Extensibility is the possibility to add functionalities to software without damaging the system; it may be thought of as a subset of flexibility. Those functionality changes may occur according to changing requirements or an obligation. Change is inevitable in software development so, this is one of the most important properties of quality software

Maintainability

Maintainability is a little similar with flexibility but it focuses on error corrections and minor function modifications rather than major functional extensibilities.

Performance and Efficiency

Performance is mostly about the response time of the software. This response time should be in acceptable intervals (e.g. max. a few seconds), and should not increase if transaction count increases. Efficiency must be supported with resource utilization. As an exaggerated example, the ability to perform a simple function only by using a 32 processor machine or 1 TB disk space is not acceptable. Optimal source/performance ratio must be aimed.

Scalability and Reliability

- A system is considered scalable when it doesn't need to be redesigned to maintain effective performance during or after a steep increase in workload. "**Workload**" could refer to simultaneous users, storage capacity, the maximum number of transactions handled, or anything else that pushes the system past its original capacity. Of course more hardware may be added for handling increasing user transaction, but the architecture should not change while doing this. This is known as vertical scalability. Reliability stands for the extent to which a program is supposed to perform its function with the required precision under workloads conditions. An unreliable system is one that is not scalable.

Availability and Fault Tolerance:

- Robust software should not lose their availability even in most failure states. Even if some components are broken down, it may continue running. Besides, even if the whole application crashes, it may recover itself using backup hardware and data with fault tolerance approaches. There should always be B and even C, and D plans.

Usability and Accessibility

- User interfaces are the only visible parts of software according to the viewpoint of the end-users. In this sense, simplicity and learnability are very important. The most well known principle for this property is KISS (Keep It Simple Stupid). Usable software should also support different accessibility types of control for people with disabilities who may find difficulties to operate input and interpret output of a program.

Portability

Portability means that software should run on as much platforms as it can. So, more people can make use of it. In different contexts we may mention different platforms: these may be OS platforms, browsers, etc.

Testability

Quality software requires quality testing. Source code should be tested with the most coverage and with the most efficient testing methods. This can be performed by using encapsulation, interfaces, patterns, or low coupling techniques correctly.

Security

Security is a very important issue in software development, especially for web or mobile based ones which may have millions of users with the ability of remote accessing to the system. You should construct a security policy and apply it correctly by leaving no entry points. This may include authorization and authentication techniques, network attack protections, data encryption and so on. All possible types of security leaks should be considered; otherwise, one day only one attack may crash your whole application and whole company.

Functionality or Correctness

Functionality or Correctness is the conformity of the software with actual requirements and specifications. In fact, this is the precondition attribute of an application, and may not be considered as a quality factor, but as the main reason for the software to exist. Quality factors are meaningless when we are talking about non-functional software. First, develop the desired functionalities and produce correct software, then apply quality factors on it.